REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE April 1996	3. REPORT TYPE AND DATES COVERED Technical Report

4. TITLE AND SUBTITLE Performance Characterization of Image Stabilization Algorithms	5. FUNDING NUMBERS DAAH04-93-G-0419
6. AUTHOR(S) Stephen B. Balakirsky and Rama Chellappa	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Computer Vision Laboratory Center for Automation Research University of Maryland College Park, MD 20742-3275	8. PERFORMING ORGANIZATION REPORT NUMBER CAR-TR-822 CS-TR-3630
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office, P.O. Box 12211 Research Triangle Park, NC 27709-2211 Advanced Research Projects Agency 3701 N. Fairfax Dr., Arlington, VA 22203-1714	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ARO 32365.8-MA

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release. Distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This paper compares three image stabilization algorithms when used as preprocessors for a target tracking application. These algorithms vary in computational complexity, accuracy, and ability. Algorithm 1 is capable of only pixel-level realignment of imagery, while Algorithms 2 and 3 are capable of full subpixel stabilization with respect to translation, rotation, and scale. The algorithms are evaluated on their performance in the stabilization of one synthetic forward looking infrared (FLIR) data set and two real FLIR imagery data sets. The evaluation tools incorporated include mean square error of the output data set and the overall performance of an automatic target acquisition system (developed at the Army Research Laboratory) that uses the algorithms as a front end preprocessor. We find that for this tracking application, extremely accurate subpixel stabilization is a requirement for proper operation. We also find that in this application, Algorithm 3 performs significantly better than the other two algorithms.

19960912 045

14. SUBJECT TERMS Image stabilization, target tracking, target acquisition, FLIR imagery	15. NUMBER OF PAGES 34
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to *stay within the lines* to meet *optical scanning requirements.*

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	Contract	PR	Project
G	Grant	TA	Task
PE	Program Element	WU	Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	See authorities.
NASA	See Handbook NHB 2200.2.
NTIS	Leave blank.

Block 12b. Distribution Code.

DOD	Leave blank.
DOE	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	Leave blank.
NTIS	Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

CAR-TR-822
CS-TR-3630

DAAH04-93-G-0419
April 1996

Performance Characterization of Image Stabilization Algorithms

Stephen B. Balakirsky
Army Research Laboratory
Adelphi, MD 20743

Rama Chellappa*
University of Maryland
College Park, MD 20742

Abstract

This paper compares three image stabilization algorithms when used as preprocessors for a target tracking application. These algorithms vary in computational complexity, accuracy, and ability. Algorithm 1 is capable of only pixel-level realignment of imagery, while Algorithms 2 and 3 are capable of full subpixel stabilization with respect to translation, rotation, and scale. The algorithms are evaluated on their performance in the stabilization of one synthetic forward looking infrared (FLIR) data set and two real FLIR imagery data sets. The evaluation tools incorporated include mean square error of the output data set and the overall performance of an automatic target acquisition system (developed at the Army Research Laboratory) that uses the algorithms as a front end preprocessor. We find that for this tracking application, extremely accurate subpixel stabilization is a requirement for proper operation. We also find that in this application, Algorithm 3 performs significantly better than the other two algorithms.

*The partial support of the Advanced Research Projects Agency (ARPA Order No. A422) and the Army Research Office under Grant DAAH04-93-G-0419 is gratefully acknowledged, as is the help of Sandy German in preparing this paper.

1. Introduction

What is image stabilization? According to Webster's New World Dictionary [1], the word "stable" means "capable of returning to equilibrium or original position after having been displaced; steady; fixed." Thus stabilization may be defined as the act of making something return to equilibrium or its original position after being displaced. When applied to imagery, stabilization refers to modifying an image sequence that comes from a moving or jittering camera so that the image appears stable or stationary.

Image stabilization technology has already made its way into the commercial marketplace. It may be found in state-of-the-art research systems such as the "VFE" from David Sarnoff Research Center and in consumer products such as home video cameras. The VFE has been used to provide stabilized video from off-road robotic vehicles, while home cameras are capable of removing high-frequency jitter which may be caused by an unsteady hand at the camera. Stabilization techniques may be broken down into two broad application areas, which involve human-observed imagery and computer-observed imagery, respectively.

Human-observed imagery refers to imagery that is intended for viewing by a human observer. This includes the home video recorder, as well as imagery that is sent back to a robotic control station (RCS). RCS imagery may be viewed to permit robotic tele-operation/tele-driving or human-assisted target acquisition/recognition. In these cases, imagery from a moving off-road vehicle may be too jittery for a human to easily or comfortably observe. Image stabilization may be used to remove this motion, and create a more viewable image for the observer. Human-observed imagery may also come from stationary cameras that are panned in order to search a large region. Precise image stabilization would provide the information necessary to allow an "image mosaic" of the entire scene to be created. In an image mosaic, the camera's entire panned region is displayed on the monitor. As new updated imagery is received from the camera, it is pasted into the appropriate region of the mosaic. This allows for continuous viewing of background contextual information around the actual live camera image.

Computer-observed imagery includes the broad class of computer vision algorithms that may be applied to an incoming video stream. This includes such widely varied applications as automatic target acquisition (ATA), automatic target recognition, autonomous mobility, and obstacle avoidance. Since a rigid computer algorithm is now "looking" at the incoming imagery, we no longer have the benefit of the human visual system as a post-processor. In fact, it will be shown that even subpixel misalignment between image frames is too much for certain ATA algorithms to handle.

This paper concentrates on the performance characterization of several stabilization algorithms with respect to how well they work in conjunction with a post-processing algorithm such as tracking. The Army Research Laboratory (ARL) has been working in the area of military robotics for many years. In cooperation with the Advanced Projects Research Agency (ARPA), ARL has fielded several computer vision systems for use in military field experiments. These platforms provide a source of known, robust algorithms in areas such as ATA. The ATA system which ARL has fielded is designed to work with imagery taken by a stable fixed camera. For this reason, it was felt that the ARL tracker performance would provide a good measure of the quality of a stabilized image sequence.

The ARL tracker output displays a box around any object which the tracker has determined to be a target vehicle. This target box has an associated target identifier with it. Under normal operating conditions (i.e., a stable input video sequence), this target identifier remains constant throughout the tracking sequence, and the surrounding box is slightly larger than the actual target (the percentage larger is a user-controlled parameter that guarantees complete target segmentation). In order for the tracker to perform the above operations, it must accurately segment the target from the background as well as be able to determine the target's motion characteristics. Image instabilities that are not compensated for will adversely affect the tracker's ability to perform these operations. Therefore, these measures, along with the time to acquire a target and the false alarm rate, will be used to judge the various stabilization algorithms.

Three different stabilization algorithms are addressed in this paper. The first algorithm (Projection Algorithm) was developed by ARL to compensate for wind loading on one of their unmanned robotic platforms. This algorithm is the least computationally expensive, and is only capable of compensating for integer image translations. The second algorithm (Feature Tracking Algorithm (FTA) 1) is a feature tracking algorithm developed by the University of Maryland. This algorithm is capable of full stabilization in translation, rotation, and scale, and provides for subpixel accuracy. This algorithm stabilizes frame n to frame $n-1$ and therefore has a very short memory. The final algorithm (Feature Tracking Algorithm 2), which is the most computationally expensive, was also developed at the University of Maryland. This algorithm is an extension of FTA 1 and requires a longer image memory.

The organization of this paper is as follows: Section 2, Algorithm Overviews, presents an overview of the three stabilization algorithms studied, as well as an overview of the tracking system. Section 3, Algorithm Characterization, provides an overview of the tracking system performance criteria as well as a definition of the performance baseline and evaluation data sets. Sections 3.3 to 3.5 discuss the results of the actual evaluations. Finally, Section 4 presents conclusions and discusses areas for future work.

2. Algorithm Overviews

2.1 Projection Algorithm

ARL implemented a real-time image stabilization algorithm for the Demo 1 robotics initiative [2]. The Demo 1 initiative consisted of five robotic high-mobility multi-purpose wheeled vehicles (HMMWVs) performing a variety of robotic missions. These included tele-operated driving, autonomous road following, autonomous retrotraverse (an automatic vehicle path retrace operation), automatic target acquisition (ATA), and automatic weapon control. The weapon control consisted of the ATA system "handing off" target locations and velocity vectors to a weapon control system which directed point fire (a 25mm cannon) and a laser designator (the moving target was illuminated with a laser beam for 5 seconds) at the selected targets.

The ARL algorithm was specifically designed to eliminate motion caused by wind loading on a stationary sensor platform. The current algorithm is capable of compensating for translational motion, and simple additions have been proposed that would allow compensation for rotation. Since the algorithm was designed for stationary platforms, no attempt was made to compensate for image scale change. The algorithm is based on the standard frame-by-frame cross correlation method [3] with modifications to allow for real-time implementation. The algorithm can be broken down into three major parts: image mapping, projection filtering, and alignment.

2.1.1 Image Mapping

Each incoming image frame is mapped from its original two-dimensional image space into two distinctive one-dimensional waveforms as shown in Figure 1. This is accomplished by using equation (2.1) to compute modified column projections. Modified row projections are computed in a similar manner.

$$Col_k(j) = \sum_i Cur_k(i, j)$$

$$ColTot_k = \left(\sum_j Col_k(j) \right) \Big/ NC$$

$$Col\,Proj_k(j) = Col_k(j) - ColTot_k \tag{2.1}$$

Here $Col_k(j)$ represents the unmodified projection of the k^{th} image, j^{th} column; $Cur_k(i,j)$ is the $(i,j)^{th}$ pixel of the k^{th} input image; NC is the number of columns; and $ColProj_k(j)$ is the modified projection of the k^{th} image, j^{th} column.

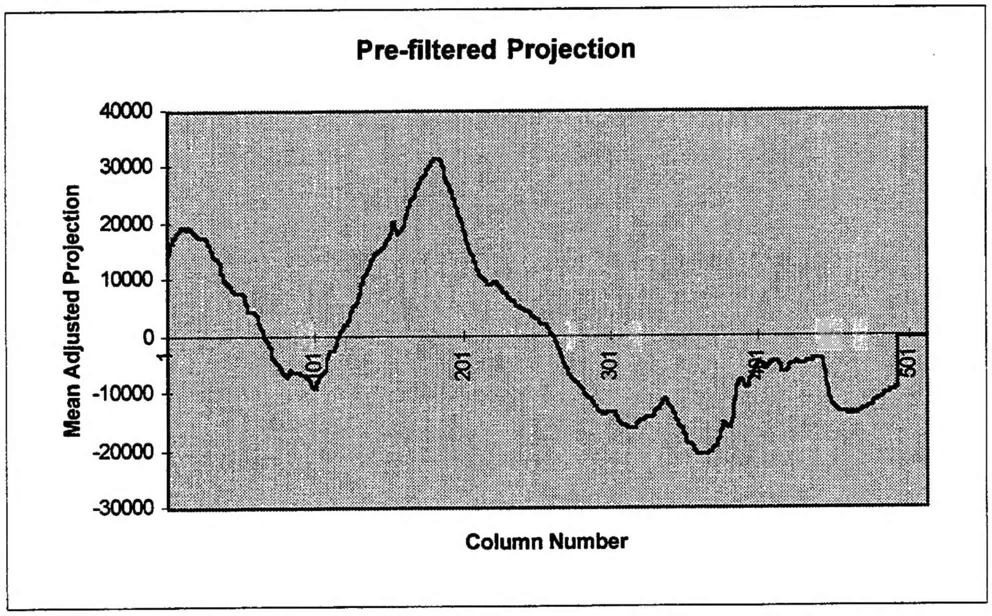

Figure 1: Projections of columns from frame 2, real data set 1.

In the standard cross-correlation technique, projections from consecutive frames would now be cross-correlated. It was found, however, that alignment accuracy could be greatly improved by filtering the projections.

2.1.2 Projection Filtering

For large image shifts (>10%), edge information is unique for each frame. It was found that this unique edge information adversely affected the cross-correlation peak. To compensate for this problem, possible projection peaks around the edges are eliminated. This is accomplished by passing each projection through the raised cosine filter given in equation (2.2). As shown in Figure 2, this filter has the effect of lowering the amplitude of the edge information while leaving the center regions intact.

$$ColProj_k(j) = \begin{cases} ColProj_k(j) \times \dfrac{1+\cos(\Pi \times (F-1-j)/F)}{2}, & j \in (0,F),(NC-F,NC) \\ 0, & \text{else} \end{cases} \tag{2.2}$$

Here F is the user-selectable filter length, and NC is the number of columns.

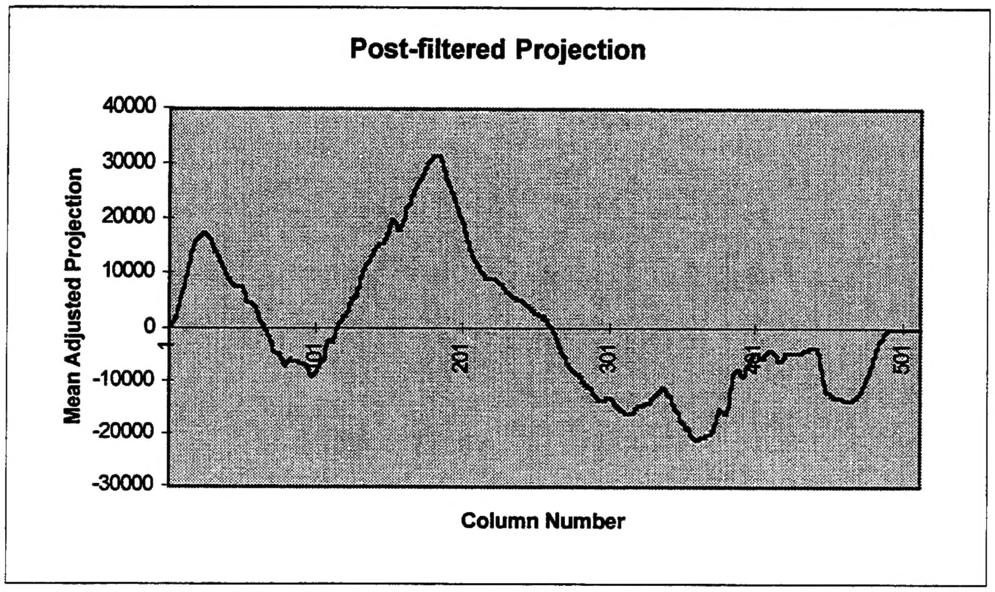

Figure 2: Filtered projections of columns, frame 2, real data set 1.

2.1.3 Alignment

A cross-correlation is performed between the waveforms of frame k and the reference image. As shown in Figure 3 this results in a unique peak location. The resulting peak is used to shift the new image into alignment. Precision may be slightly improved by performing a regional two-dimensional cross-correlation around the intersection of the highest row and column peak. A rotational correction step was designed, but not implemented, for this algorithm. This step involved correlating the region around the highest row/column peak with warped (in rotation) image chips from the reference image. The highest cross-correlation value would dictate the amount of rotation.

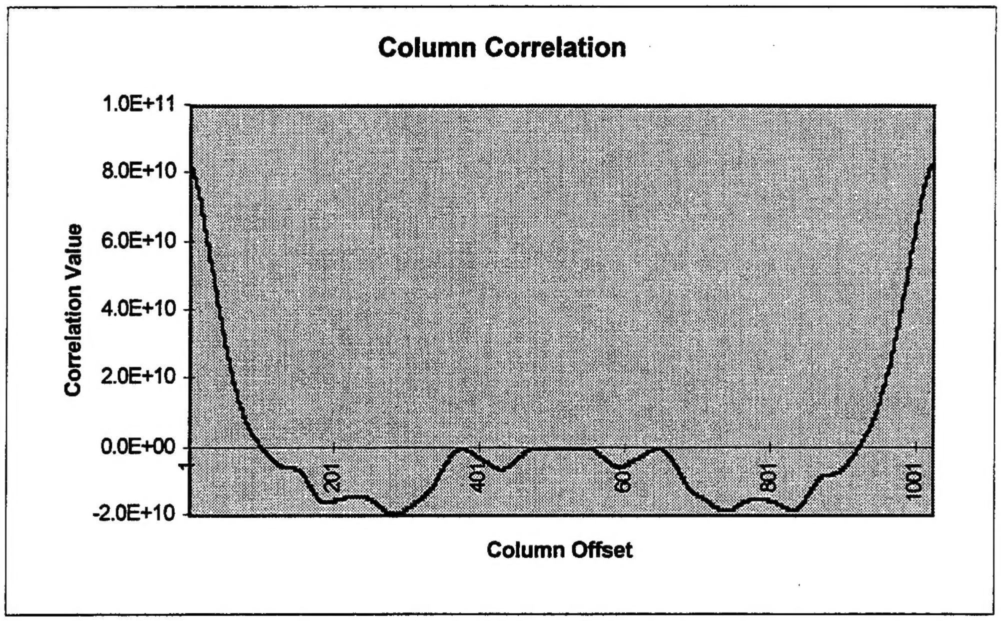

Figure 3: Cross-correlation of frame 1 with frame 2.

2.2 FTA 1

The first feature tracking algorithm was one developed by Qinfen Zheng [4] and modified for real-time implementation on Datacube hardware by Carlos Morimoto et al. [5] at the University of Maryland. This algorithm is a feature-based algorithm which operates on a hierarchical image set through the use of a Gaussian pyramid. The current real-time implementation accepts NTSC video input and displays its stabilized output at a final resolution of 128×128 pixels. This algorithm is capable of compensating for image translation, scale, and rotation. The magnitude of motion which may be compensated for is directly related to the coarsest pyramid resolution and correlation search space (see Section 2.2.2 on Feature Detection). In order to characterize this algorithm, the Datacube software was ported to a Sun station and modified to have a final output resolution of 512×512.

The steps of this algorithm are as follows:

1. Create Gaussian pyramid from image.
2. Detect and select a small set of features (not performed for every image).
3. Match area surrounding current features with previous frame.
4. Compute estimate of image motion based on matched feature displacement.
5. Transform next higher level of pyramid by current estimate of image motion.
6. Repeat from step (3) for all levels of pyramid until reaching final output resolution.

2.2.1 Gaussian Pyramid

FTA 1 is a multi-resolution approach to image stabilization. In order to extend the range of the motion estimation process, a multi-resolution Gaussian pyramid structure is implemented. Successive levels of this pyramid structure "shrink" the resolution and sample density of the original image by powers of two. This has the effect of increasing the frame-to-frame motion which may be compensated for. For example, in a three-level pyramid, a shift of one pixel corresponds to a four-pixel shift in the original image. By successively solving the fine-motion problem at each level of the pyramid, and applying the intermediate results to warp the original image, one can bring the original image into fine alignment. If we define the sample distance as the largest pixel shift which may be compensated for, and assume that the image motion lies in this range, then this successive alignment procedure ensures that the residual displacement between levels (amount of displacement which has yet to be determined) is less then the sample distance [6].

Formally, if we let $G_{t,n}$ be the n^{th} pyramid level for the image $I(x,y,t)$, we may form subsequent levels of the pyramid by convolving the previous level with a kernel filter ω and sub-sampling [7]:

$$G_{t,n} = \left[G_{t,n-1} \otimes \omega\right]_{\downarrow 2} \tag{2.3}$$

where ↓2 indicates that the quantity in brackets has been sub-sampled (both rows and columns) by a factor of two. This technique has been successfully used to estimate motion at accuracies of a small fraction of a pixel [8, 9, 10].

2.2.2 Feature Detection

For the purpose of this algorithm, a feature is defined as a point in the image which represents a large contrast change. Object edges offer an excellent source of such contrast changes. In order to find such edges, all levels of the Gaussian pyramid are convolved with a Laplacian mask to create an edge image. The Laplacian mask acts as an edge detector by highlighting areas of sharp contrast change while suppressing areas of uniform contrast [11]. The pyramid level which represents the resolution of the final stabilized image is next broken down into a number of equal-width columns. Each of these columns is searched for the pixel highest in value, and this pixel is marked as a feature point.

This method of feature extraction was selected because of its ease of implementation in a real-time system. It is a very simplistic approach to feature extraction, that is designed to locate areas of the image which contain sharp, well-defined edges. The actual point selected as a feature point is not important, and may change from frame to frame. What is important is that the area, or window, surrounding the feature point contain an area of high contrast. In the next step of the algorithm, the feature point will be translated down through the pyramid into the coordinate space of the finest pyramid level. A windowed, weighted correlation and subpixel matching algorithm will then be performed on the edge image in order to determine the extent of the image motion. It is this windowed correlation that determines the maximum image movement which may be compensated for. It is necessary for the true correlation peak to lie within the window of the coarsest pyramid level. If this is the case, subsequent pyramid levels will then be used to refine the estimate. This means that increasing either the number of pyramid levels or the size of the correlation search window will result in being able to compensate for larger image motions.

2.2.3 Area Matching

The area matching algorithm operates over a three-frame interval. Feature areas from the previous or next frame are matched with the current frame, and motion estimates are obtained. Since this bases future stabilization estimates on the current estimate, stabilization errors may additively propagate through the algorithm, causing a noticeable drift in the stabilized image. This image drift is caused by both quantization error and error due to false peak selection in the correlation output [12]. To compensate for the possible drift incurred through the use of the cross-correlation technique, this algorithm uses a two-step, subpixel-accuracy matching algorithm. If each pixel is viewed as a grid point in frame n, then a match to the corresponding grid point in frame $n-1$ is obtained by using a weighted cross-correlation match. The match is then refined by using a differential method to achieve subpixel accuracy [12, 13]. Finally, these matched feature points are used to calculate the actual image motion.

2.2.3.1 Weighted Correlation

There is a tradeoff in the use of a correlation match. A larger area used in computing the correlation causes better selectivity over similar features, but less accuracy in the actual location. To obtain high location accuracy, while rejecting false peaks caused by similar features, a weighted correlation scheme is used. This correlation uses a large, symmetrically weighted window γ which has high values in the center and decreasing values as the edges are approached. The large area yields good selectivity, and the weighting maintains good feature localization. The modified matching criterion is [12]

$$\Psi_{\mathbf{f}_1,\mathbf{f}_2}(m,n,u,v) = \frac{\sum_{i,j}\left(\gamma_{ij} \times \mathbf{f}_1(m+i,n+j) \times \mathbf{f}_2(u+i,v+j)\right)}{\sum_{i,j}\gamma_{ij}\mathbf{f}_1^2(m+i,n+j) \times \sum_{i,j}\gamma_{ij}\mathbf{f}_2^2(m+i,n+j)},$$

where

$$\gamma_{ij} = \frac{\gamma_{00}c}{\left(\max\{i,j\}\right)}, \qquad \{i,j\} \neq \{0,0\},\ \mathrm{c} = \text{constant} \tag{2.4}$$

2.2.3.2 Subpixel Matching

It is assumed that the above correlation technique provides an accurate match to within ±0.5 pixels in each direction. This estimate of motion is further refined using the following subpixel matching algorithm:

Assume that frame $\mathbf{f}_2$ is offset from frame $\mathbf{f}_1$ by $(\delta x, \delta y)$; then

$$\mathbf{f}_2(i,j) = \mathbf{f}_1(i-\delta x, j-\delta y) \tag{2.5}$$

The frame difference may be written as follows:

$$\begin{aligned} d(i,j) &= \mathbf{f}_1(i,j) - \mathbf{f}_2(i,j) \\ &= \mathbf{f}_1(i,j) - \mathbf{f}_1(i-\delta x, j-\delta y) \\ &\approx \frac{\partial \mathbf{f}_1(i,j)}{\partial x}\delta x + \frac{\partial \mathbf{f}_1(i,j)}{\partial y}\delta y \end{aligned}$$

(2.6)

where the derivatives of $\mathbf{f}_1$ may be approximated by forward differences. Examining a small neighborhood around the feature point yields a set of simultaneous equations that may be solved to find the values of δx and δy. Keeping this neighborhood small allows for reduced computations as well as achieving better localization [14].

2.2.4 Motion Estimation

From the set of matched feature pairs, we must now derive information that will allow us to warp the current image frame into alignment with the previous image frame. It is therefore necessary to compute the scale and rotation change, and the horizontal and vertical translation. These are represented by $s', \theta', \Delta X,$ and ΔY respectively. The first parameter to be computed is the scale. Since the Euclidean distance between two feature points is invariant to changes in translation and rotation, the scale may be computed as

$$s' = \sum_{i=1}^{N'} d_i * d_i' \bigg/ \sum_{i=1}^{N'} d_i * d_i$$

(2.7)

where N' is the number of matched feature points, and d_i, d_i' are the distances from the feature point to the center of gravity of all of the matched feature points in frames t_1 and t_2, respectively.

If we limit our frame-to-frame rotation to be small, then $\sin\theta$ and $\cos\theta$ may be approximated in linear terms. The solution for the translation and rotation then becomes the solution of a set of linear equations:

$$\begin{pmatrix} \hat{X}_i \\ \hat{Y}_i \end{pmatrix} = \begin{pmatrix} 1 & \theta' \\ -\theta' & 1 \end{pmatrix} \begin{pmatrix} s'X_i' \\ s'Y_i' \end{pmatrix} + \begin{pmatrix} \Delta X' \\ \Delta Y' \end{pmatrix}, \ i = \{1, \ldots, N'\}$$

(2.8)

where $(\hat{X}_i, \hat{Y}_i)$ and (X_i', Y_i') are the coordinates ofthe matched feature points in frame t_2 and the transformed frame t_1, respectively, and $s', \theta', \Delta X',$ and $\Delta Y'$ are the scale, rotation, and translation parameters which need to be estimated. Solving equations (2.7) and (2.8) for these parameters provides all the information necessary to transform the current image frame on a pixel-by-pixel basis through an affine transform to match the location of the previous frame.

2.3 FTA 2

FTA 1 was designed to work on human-observed imagery from a mobile off-road vehicle. As such, this algorithm strives to align frame *n* to frame *n*+1. This results in the system having a very short memory. For the case of the ARL tracker, a very long system memory is desirable (i.e., aligning frame *n* to the reference frame). In order to accommodate this, FTA 1 was modified to perform alignment of each incoming frame with the reference frame. The steps of the modified algorithm are as follows:

1. Create Gaussian pyramid from image.
2. Detect features of image (not performed for every image).
3. Match area surrounding current features with previous frame.
4. Compute estimate of image motion based on feature motion.
5. Transform next highest level of pyramid by current estimate of image motion.
6. Repeat from step (3) for all levels of pyramid until reaching final output resolution.
7. Match area surrounding reference frame features with current transformed frame.
8. Compute residual motion of image based on feature motion
9. Transform current image by refined motion estimation.

It should be noted that the only change to the algorithm is the addition of steps 7 through 9. These steps provide additional refinement and correct for any residual drift that was not removed by the subpixel matching phase.

2.4 ARL Tracking System

The ARL automatic target acquisition (ATA) system was developed as an integrated multi-sensor (FLIR and acoustic) approach to tracking multiple moving targets from a stationary platform. This algorithm, developed for the Technology-based Enhancements for Autonomous Machines (TEAM) program, is currently being used by the Robotics Demo II program. This tracking algorithm attempts to overcome many of the classic ATA problems, such as tracking in cluttered environments, correctly distinguishing between multiple overlapping targets, and discriminating between actual moving vehicles and "false" moving objects (dust clouds, small animals, waving tree branches) [2]. Most ATA systems follow the following steps in processing an incoming video stream [15]:

1. Signal preprocessing is performed to improve target contrast and reduce sensor noise.
2. Potential targets are located.
3. Potential targets are segmented from the background.
4. Features of each potential target are used to discriminate real targets from non-targets.

The ARL approach to ATA fits into the above paradigm.

2.4.1 Signal Preprocessing

In order for the ATA system to recognize the presence of targets, it must first have a target-free view of the world. We call this image the "reference image" because it represents a target-free baseline for the system. This image is acquired at system setup time and is allowed to slowly change to reflect changing conditions in the tracking area (i.e., changes in illumination). Targets, however, must not be allowed to become part of this reference image. If they were present, the ATA system would include them in the image baseline and would not continue to recognize them as possible targets. By slowly updating the reference image, the ATA system is given time to recognize target areas before they become part of the image baseline. The reference image is then selectively updated:

$$Ref_{k+1}(i,j) = \begin{cases} Ref_k(i,j) & \text{if there is a target at pixel}(i,j) \\ \alpha * Ref_k(i,j) + (1-\alpha) * Cur_k(i,j) & \text{otherwise} \end{cases} \tag{2.9}$$

where $Ref_k(i,j)$ is the $(i,j)^{th}$ pixel of the k^{th} reference image, $Cur_k(i,j)$ is the $(i,j)^{th}$ pixel of the k^{th} input image, and $\alpha\,(0 \le \alpha \le 1)$ determines the rate at which the reference image is updated. Whether or not there is a target at a particular pixel is determined by the ATA system's current view of target locations.

A bandpass filter is next applied to a difference image which is formed by using equation (2.11). This filter detects differences in the "structure" of the input images which point to likely target locations. The high-pass section of this filter reduces the effects of broad illumination changes, while the low-pass section reduces the effects of noise. This structural difference (*SDiff*) is calculated by

$$SDiff_k(i,j) = \frac{1}{N_1^2} \left| \sum_{m,n \in R_1} \left(Diff_k(i+m, j+n) - \overline{Diff_k(i+m, j+n)} \right) \right| \tag{2.10}$$

where

$$Diff_k(i,j) = Ref_k(i,j) - Cur_k(i,j) \tag{2.11}$$

and

$$\overline{Diff_k(i,j)} = \frac{1}{N_2^2} \sum_{n,m \in R_2} Diff_k(i+m, j+n) \tag{2.12}$$

R_1 and R_2 are regions centered around (i,j) and are of user-defined size. They are typically bounded by $\left(-N_p/2, N_p/2 - 1\right)$, $p = (1,2)$ where N_1 and N_2 have the values 4 and 8, respectively.

2.4.2 Locating Potential Targets

If one examines the *SDiff* image over time, certain noisy regions of the image (where noise is due to local motion such as moving tree limbs) will consistently display higher values than quiet areas. Therefore, a consistent threshold for considering an area to be "interesting" may give false alarms on noisy areas while missing real targets over quiet areas. To deal with this phenomenon, a dynamic noise image ($Noise_k$) is used. In order for a given area to be considered "interesting," it must now be greater than the noise image by a user-given threshold δ. This noise image has a different value for each pixel and is determined by

$$Noise_{k+1}(i,j) = \begin{cases} Noise_k(i,j) & \text{if target at pixel}(i,j) \\ \beta * Noise_k(i,j) + (1-\beta) * SDiff_k(i,j) & \text{otherwise} \end{cases} \tag{2.13}$$

Once again, whether or not there is a target at pixel *(i,j)* is determined by the ATA system's current view of the target locations, and β $(0 \le \beta \le 1)$ is user-defined. A binary image ($Targ_k$) is then formed by

$$Targ_k(i,j) = \begin{cases} 1 & \text{if } SDiff_k(i,j) - Noise_k(i,j) > \delta \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

where δ is a user-defined parameter. In order to make further calculations tractable (the object of this system is to work in real time), $Targ_k$ is sub-sampled on a 2D grid of evenly spaced points. This sub-sampling reduces the amount of remaining computation without significantly affecting the system's ability to detect targets.

2.4.3 Target Segmentation

The tracking system next segments the image that results from the sub-sampling by extracting connected components [16]. The segmentation creates a set of symbolic objects which represent areas of the image which are changing, and further reduces the image area which must be examined to locate targets. Each of these objects has an associated set of parameters which include object shape, centroid, area, and range. These objects are next examined to discriminate between changes due to noise and clutter, and changes due to true targets.

A feature which is not yet available in the real-time system, but which may be run in the computer simulation, makes an attempt to separate two or more targets that may be overlapping by performing an optical-flow-like algorithm [17]. To perform this analysis, a 2D grid of points is overlaid on top of the object in question. Next, a velocity vector is computed for each of these grid points. These velocity vectors are grouped into areas of uniform motion, and the object is then resegmented.

The velocity vector is computed by determining the motion of a small neighborhood of points centered on each grid point. This computation is accomplished by performing a template matching between regions in the current image and the previous image:

$$Vel_a = \min_{l,m \in R_1} \left(\sum_{i,j \in R_2} \left(Cur_k(i,j) - Cur_{k-1}(i+l, j+m) \right)^2 \right)$$

(2.15)

Equation (2.15) attempts to minimize the sum of squared differences over a region R_1. Region R_1 is determined by the amount a target in region R_2 would move in a single frame time. Once the velocities for each grid point are computed, similar adjacent velocity vectors are grouped into regions of uniform motion. These regions of uniform motion are then used to resegment the set of objects which was previously generated. This resegmentation will break apart overlapping targets that are moving in different directions or at different speeds. It also helps to eliminate high-clutter areas since they will usually exhibit random motion.

2.4.4 Target Discrimination

The above algorithms generate a set of potential targets. The tracker must decide which of these objects represent true targets and which are the results of noise and clutter. A survey of various multi-target tracking algorithms is given in [18]. The ARL tracker tracks each object over several frames before the determination of "target" or "clutter" is made. This allows the system to make a more informed decision, and allows for better estimation of object properties.

The fundamental problem which must be overcome is that of determining the frame-to-frame correspondences of objects. ARL uses a best-first search [19] algorithm to determine an optimal frame-to-frame correspondence of the objects. Target properties, such as velocity and range, are used to limit the set of possible correspondences. Additionally, multiple objects may be matched to a single previous object. This allows for the fact that poor segmentation may cause an object to "break apart" into multiple objects. The goodness of any particular correspondence is based on the weighted differences of the object properties. The result of the correspondences is an updated list of possible targets. Each of these objects' properties is then examined to determine if the object is exhibiting "target-like" behavior. Factors such as object permanence, velocity, and size are examined. A violation in any one area is enough to rule out an object's being a target. If the object passes these tests, it is added to the "target" list. Objects on the target list are displayed on the tracker's output by surrounding them with white boxes, while areas which exceed the motion threshold are shown by having black boxes overlaid on them. A typical output frame from the tracker is shown in Figure 4.

3. Algorithm Characterization

The goal of this paper is to evaluate stabilization algorithms of widely varying computational complexities in a way that has meaning in the real world. The classic sum of mean square errors (MSEs) is used to show that one algorithm offers better performance in the mean squares sense than another algorithm. But what does better stabilization in the mean square sense mean to a real-time imaging application? How much error is acceptable? Is the added expense and complexity which would be necessary to implement the more computationally expensive algorithms necessary or worth it? To perform such an evaluation, a real-time image processing system needed to be chosen as the evaluator. For the purpose of this paper, the ARL real-time tracker was chosen as the evaluator. The tracking system was run on the output of each stabilization algorithm, and its relative performance was monitored. The relative tracker performance was then used to determine which algorithms were "good enough" to be used as a front end system to provide image stabilization for the tracker.

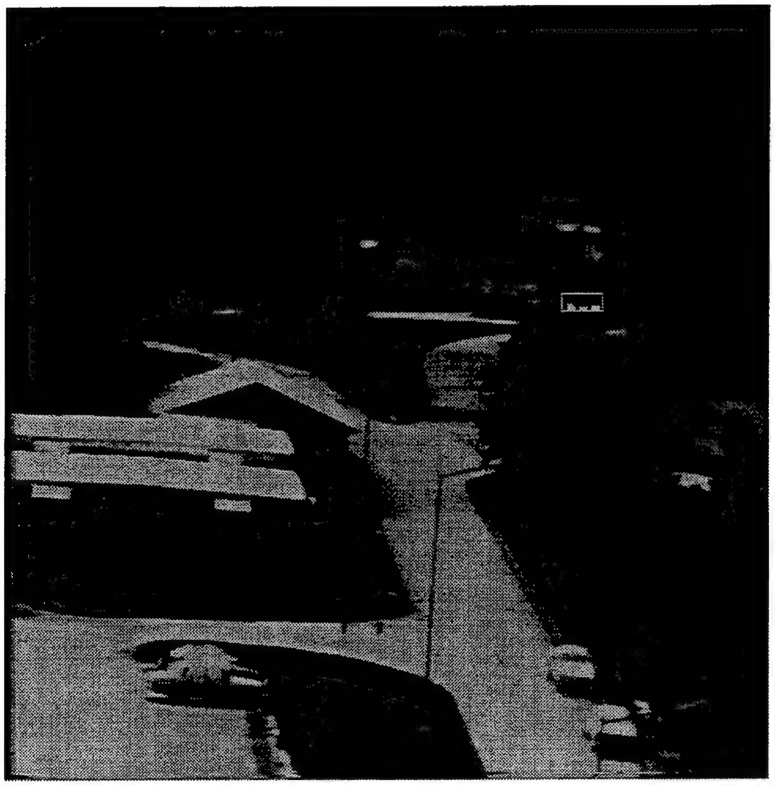
Figure 4: Typical tracker output.

3.1 ARL Tracker Performance Criteria

The output of the ARL tracking system consists of five parameters: target number, target height, target width, target velocity, and target centroid. During proper operation, the target number remains constant from track initiation until the target leaves view and is used to identify the remaining parameters. The target height, target width, target velocity, and target centroid are used by applications that use the tracking system as a front-end preprocessor. These systems include weapon control systems and automatic target recognition systems. For the purpose of this evaluation, it was decided to evaluate tracker performance based on the accuracy of these parameters, along with false alarm rate and time to acquire target from first sighting. The baseline for this evaluation was created by observing tracker performance on a sequence obtained from a stationary forward looking infrared imager (FLIR). One synthetic non-stable FLIR image sequence and two real non-stable FLIR image sequences were then run through each stabilization algorithm for evaluation. Both real sequences used for evaluation were 100 frames in length, taken at a frame rate of ten frames per second.

Table 1: Data Set 1 Image Transform Coordinates

	Absolute Shift			Shift From Frame *N*– 1				Absolute Shift			Shift From Frame *N*– 1		
Frame #	Row	Col	Angle	Row	Col	Angle	Frame #	Row	Col	Angle	Row	Col	Angle
1	0.0	0.0	0.0	-	-	-	19	11.0	15.0	0.0	5.0	5.0	0.0
2	0.1	0.0	0.0	0.1	0.0	0.0	20	17.0	9.0	0.0	6.0	-6.0	0.0
3	0.3	0.0	0.0	0.2	0.0	0.0	21	10.0	2.0	0.0	-7.0	-7.0	0.0
4	0.7	0.0	0.0	0.4	0.0	0.0	22	2.0	-6.0	1.0	-8.0	-8.0	1.0
5	2.0	0.0	0.0	1.3	0.0	0.0	23	2.0	-6.0	3.0	0.0	0.0	2.0
6	4.0	0.0	0.0	2.0	0.0	0.0	24	2.0	-6.0	6.0	0.0	0.0	3.0
7	7.0	0.0	0.0	3.0	0.0	0.0	25	2.0	-6.0	10.0	0.0	0.0	4.0
8	11.0	0.0	0.0	4.0	0.0	0.0	26	2.0	-6.0	5.0	0.0	0.0	-5.0
9	16.0	0.0	0.0	5.0	0.0	0.0	27	2.0	-6.0	-4.0	0.0	0.0	-9.0
10	10.0	0.0	0.0	-6.0	0.0	0.0	28	2.0	-6.0	3.0	0.0	0.0	7.0
11	3.0	0.0	0.0	-7.0	0.0	0.0	29	2.0	-6.0	11.0	0.0	0.0	8.0
12	-5.0	0.0	0.0	-8.0	0.0	0.0	30	2.0	-6.0	2.0	0.0	0.0	-9.0
13	-14.0	0.0	0.0	-9.0	0.0	0.0	31	2.0	-6.0	-8.0	0.0	0.0	-10.0
14	-4.0	0.0	0.0	10.0	0.0	0.0	32	2.0	-6.0	1.0	0.0	0.0	9.0
15	-3.0	1.0	0.0	1.0	1.0	0.0	33	2.0	-6.0	3.0	0.0	0.0	2.0
16	-1.0	3.0	0.0	2.0	2.0	0.0	34	4.0	-4.0	6.0	2.0	2.0	3.0
17	2.0	6.0	0.0	3.0	3.0	0.0	35	7.0	-1.0	10.0	3.0	3.0	4.0
18	6.0	10.0	0.0	4.0	4.0	0.0	36	11.0	3.0	5.0	4.0	4.0	-5.0

The baseline FLIR sequence was obtained from a FLIR imager mounted on a fixed tripod. During this image sequence, five targets are observed. The synthetic image sequence consists of a single background image which has a target image superimposed on it. The generated sequence is then operated on by an affine transform to simulate image instability. Translations as well as rotations are simulated as shown in Table 1. The simulated image shifts ranged in value from subpixel shifts to instantaneous shifts of 8 pixels in both the row and column directions. Rotational shifts were simulated in a range from 1 to 10 degrees. Finally, both shifts and rotations were simulated simultaneously. Real sequence Nos. 1 and 2 consist of real FLIR imagery collected from a tripod-mounted FLIR imager.

For real sequence No. 1, the imager was panned horizontally and vertically to obtain image rotations around the x and y axes. During these image rotations, five targets are observed. For real sequence No. 2, the imager was panned horizontally and vertically, as well as rotated. This allowed for the collection of data that contained image rotations around all three image axes. During real sequence No. 2, four targets were observed.

3.2 Performance Baseline

The MSEs between frames and between frame 1 and the current frame for the baseline image sequence are shown in Figure 5. For the purpose of this paper, MSE is defined as the square root of the sum of the squares of the pixel differences in the image set:

$$MSE_k = \sum_{i,j} \sqrt{\left(I_{1k}(i,j) - I_{2k}(i,j)\right)^2} \tag{3.1}$$

While MSE is not the measure by which this paper is judging stabilization algorithms, it does offer some insight into the stability of the image sequence. The MSE between frames may be viewed as a measure of how rapidly the image sequence is changing, and the MSE between frame 1 and the current frame may be viewed as a measure of how much the current image has drifted away from the first or reference image. As would be expected for a stable sequence, the MSE remains relatively constant, with the frame-to-frame MSE slightly lower than the reference frame to current frame MSE.

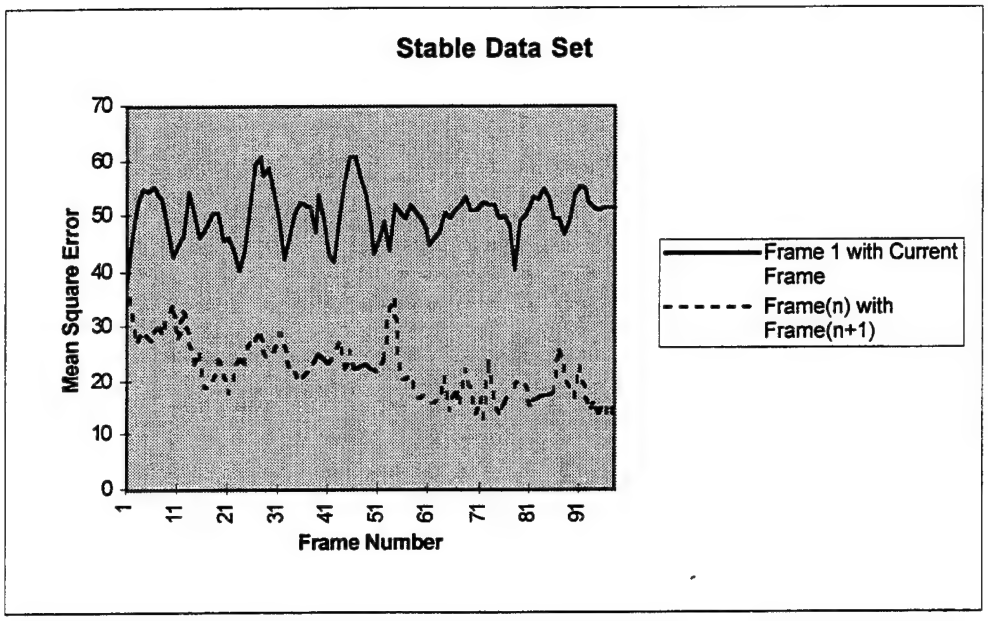

Figure 5: Baseline MSE performance.

During the image baseline sequence, five targets were observed. The average time for a target to be detected and tracked was 1.5 frames from full target view (frame when the target comes into full view of the tracker) or 5 frames from first target appearance (frame when any part of the target becomes visible). The reason for the short time between full target view and target tracking is that the tracker has already started to anticipate a target track by the time the target is in full view. The tracker maintained a consistent track ID on all targets throughout the target visibility period, and maintained a target box that was larger than the true target size of the target by at most 60%, with the centroid of the box on the centroid of the target. The tracker normally maintains a tracking box that is larger than the predicted target size to ensure that the full target is segmented and passed to other applications. Note that on the baseline run, the tracker performed a "target hand-off," a phenomenon that sometimes occurs when one target is exiting a region of the scene as another is simultaneously entering. The tracker may then mistakenly hand off the target id of the exiting target to the new target.

The tracking threshold for this image set was set at a value of 4. The tracking threshold, an important measure of tracking sensitivity, represents the average change that must occur in an image region for a change detection alarm to be triggered. Misaligned images will tend to appear noisier due to frame-to-frame image motion. This necessitates raising the tracking threshold and thus lowering the tracking sensitivity.

3.3 Results From Synthetic Data Set 1

As shown in Figure 6, the between-frame MSE is low for all three algorithms until the instantaneous row and column shift is greater than 4 (frame 18, see Table 1). At this point, the projection method breaks down and is no longer able to correct for any of the larger shift values. Both feature tracking algorithms perform very well on all shift sequences tried.

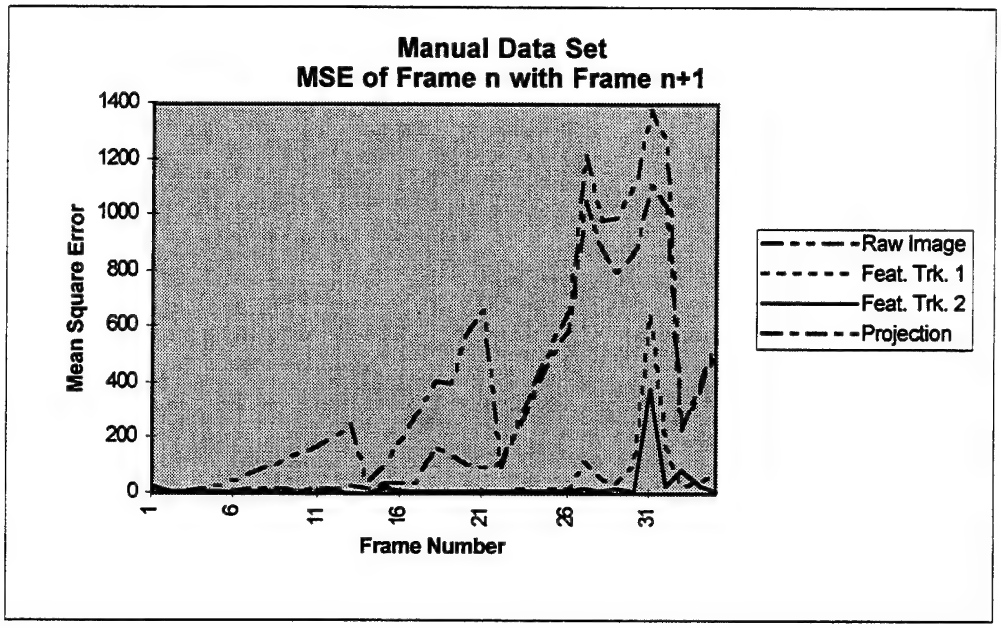

Figure 6: Synthetic data set 1, MSE frame *n* to frame *n*+1.

The projection method immediately breaks down when it is faced with trying to compensate for image rotations. As shown in Figure 6, this method provides a very slight improvement in MSE over the raw image sequence. Once again, both feature tracking algorithms perform well when compensating for both rotations and simultaneous translation/rotation. Both break down for instantaneous rotations of over 9–10 degrees.

Figure 7 shows the MSE of the current frame compared to the reference frame. Note that both the projection and feature tracking 2 algorithms exhibit very low MSE until the algorithms break down. Feature tracking algorithm 1 exhibits a noticeable drift in the MSE. This drift indicates that the subpixel matching is not fully compensating for all of the error and that the error is accumulating.

The next item examined was the actual tracker performance on each of the stabilized sequences which is shown in Table 2. For all the data sets, the tracker was run with a threshold of 9, which gave the best level of detection while eliminating false alarms.

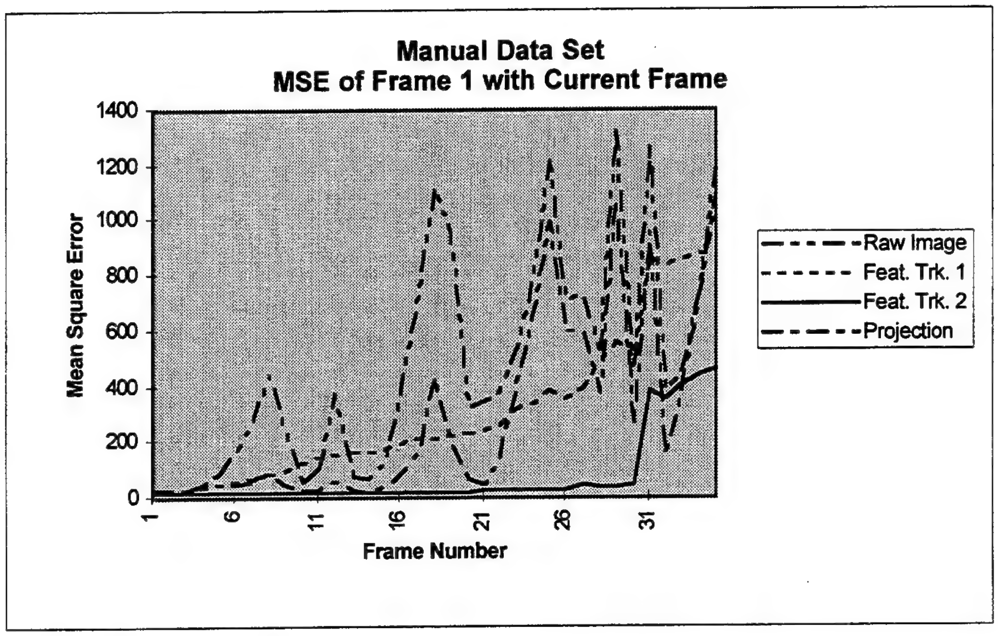

Figure 7: Synthetic data set 1, MSE reference frame to current.

Both the projection algorithm and FTA 2 experienced no false alarms while they maintained valid tracks. When examined over the entire sequence, FTA 1 experienced an average of one false alarm per frame. Appendix A shows the actual error in each of the algorithms. Even though this appendix shows that FTA 2 performed the best in stabilizing the image sequence, the projection algorithm appears to allow for better segmentation through frame 16. When the tracker was run on the output of both FTA 1 and FTA 2, it appears that the single target was incorrectly segmented into two objects. However, appearances may be deceiving. This apparently incorrect segmentation is due to parts of the target being the same temperature as the background through which the target is traveling. With proper stabilization, the separated target regions appear to be two separate entities. When the projection algorithm was used, there remained enough uncorrected motion that the two motion regions blurred into a single, correctly sized region.

Table 2: Tracker performance on synthetic data set 1.

Algorithm	First Frame Target Tracked	Loses Segmentation	Loses Track
Projection	6	14	21
FTA 1	7	32	32
FTA 2	7	13	24

3.4 Results From Real Data Set 1

Real data set 1 contains imagery with rotations around the *x* and*y* axes. Seen in Figure 8 all three algorithms performed with approximately the same MSE when frame *n* is compared to frame *n*+1. The main distinguishing characteristic is that the projection algorithm suffered from several grossly misaligned frames (as seen from the large spikes in the graph).

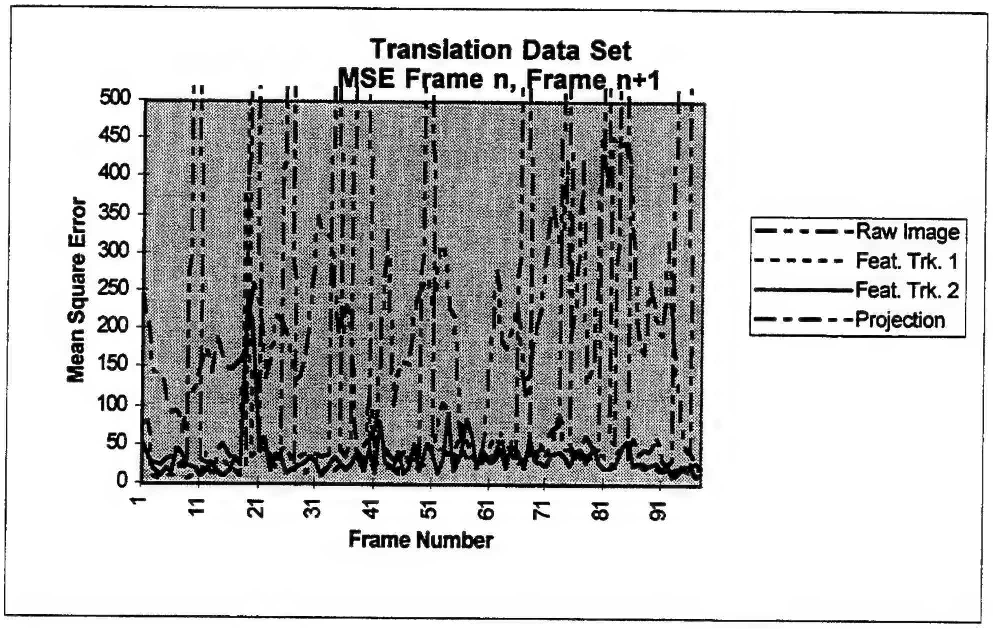

Figure 8: Real data set 1, MSE, frame *n* to *n*+1.

Figure 9 shows that the projection algorithm also suffered misalignment from the reference frame in several regions. In the early part of the image sequence, this misalignment did not adversely affect the tracker performance. This may be the result of the tracker's adaptive reference image. Since the early projection errors presented a step function, the reference image was able to adapt to the new level and the tracker was able to continue to perform well. However, later in the sequence, the projection algorithm drifted out of alignment. This drift presented the tracker with a stabilized input that had a ramp in addition to the step present in the MSE (see Figure 9, frame 37 on). The tracker was unable to perform in the presence of this MSE ramp. When the tracker threshold was set to a value of 17, the tracker experienced no false alarms until frame 39 (a point slightly after the beginning of the ramp). After that point, the tracker averaged 1.5 false alarms per frame and had difficulty in maintaining consistent track identifiers on the targets.

FTA 1 also had a problem with drift in the stabilized image sequence. This drift may be seen as the ramp in the MSE of Figure 9. In fact, this drift made the tracking system unable to use this sequence as input. As shown in Figure 10, when the tracking threshold was set low enough to detect the vehicle's motion, the entire roadway showed up as a motion region. This prevented the tracker from performing proper segmentation and establishing a track. When the threshold was raised enough to prevent the roadway from being tagged as a motion region, the vehicle failed to be detected. In both cases, numerous false alarms were generated.

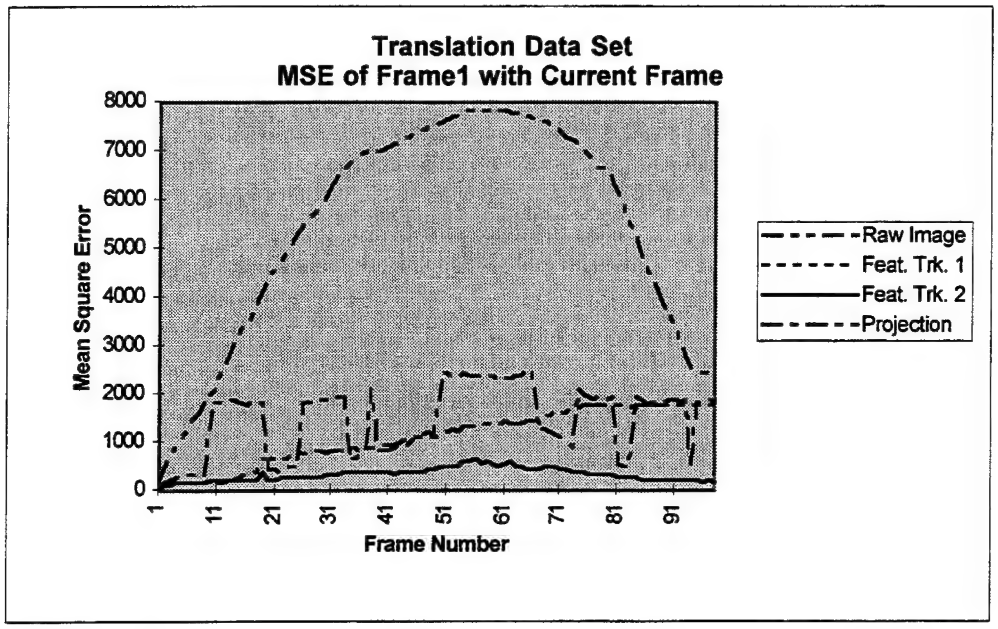

Figure 9: Real data set 1, MSE current frame to reference frame.

(a) Output of tracker, threshold 15. (b) Output of tracker, threshold 20.

Figure 10: FTA 1 performance on real data set 1.

FTA 2 allowed the tracker to perform very well against this data set. In fact, when the algorithm was running at a threshold of 12, every target was detected within 5 frames, and the average time to detect a target from full view was 3.3 frames. Lower thresholds increased the false alarm rate without significantly improving the target detection speed or segmentation. Over the entire 100-frame data set, only two frames contained a single false alarm each. The segmentation resulting from this algorithm was good, although the target box was on average only 80% of the size of the actual target. This size may be accounted for by the higher threshold, which was necessary. This higher threshold lessens the sensitivity of the tracker, and could cause it to miss faint, outer edge target pixels. Table 3 summarizes the tracker performance for this data set.

Table 3: Tracker performance on real data set 1.

Algorithm	Threshold	% targets detected	Avg. # of frames to acquire target	Avg. false alarms/frame	% targets segmented
Projection	17	100	3.4	0.93	74
FTA 1	15	0	NA	3.10	0
FTA 2	12	100	3.3	0.00	80

3.5 Results From Real Data Set 2

Real data set 2 involved imagery that had rotations around all three image axes. As may be seen from Figure 11, all the algorithms performed inconsistently with respect to frame-to-frame MSE. Figure 12 shows significant drift in all the algorithms. The tracker was unable to track any targets on the output from the projection algorithm and FTA 1. In fact, the projection algorithm failed to provide any noticeable (to the human eye) improvement in the image sequence. Performance on the output of FTA 2 was the best of the three algorithms characterized. With a tracking threshold of 12, the tracker required between 1 and 13 frames with an average of 7 frames to acquire targets once they were in full view. Although the tracker was able to detect and track all the targets, segmentation was poor, with an average of only 67% of the targets being segmented. False alarms were the main problem encountered in using the output of FTA 2, with a maximum of five false alarms in a single frame and an average of one false alarm per frame. These results are summarized in Table 4.

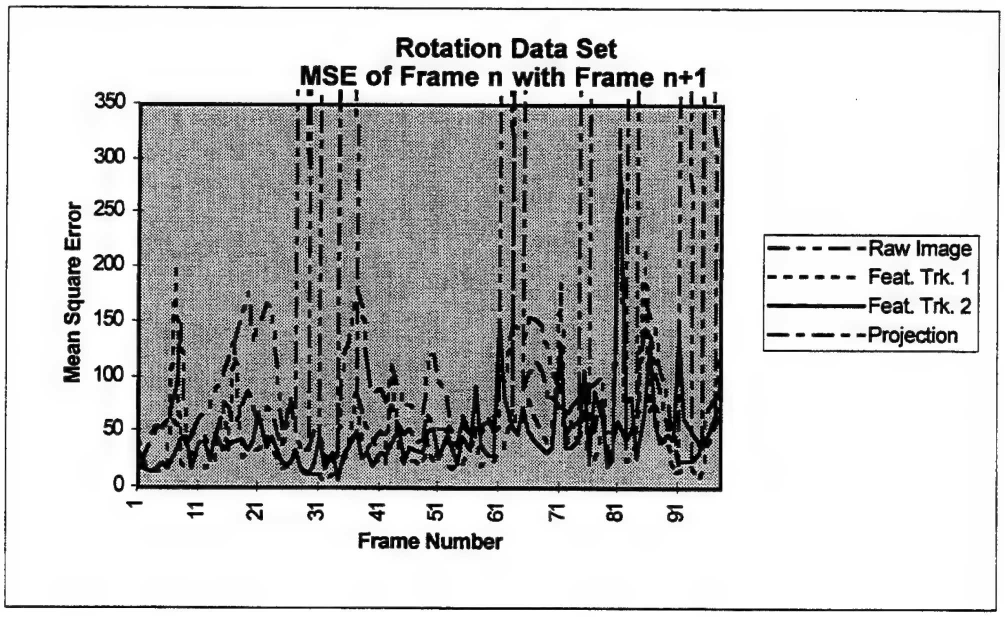

Figure 11: Real data set 2, MSE frame n to frame $n+1$.

Table 4: Tracker performance on real data set 2.

Algorithm	Threshold	% targets detected	# frames to acquire target	Average false alarms/frame	% targets segmented
Projection	17	0	NA	NA	NA
FTA 1	12	0	NA	NA	NA
FTA 2	12	100	7	1	67

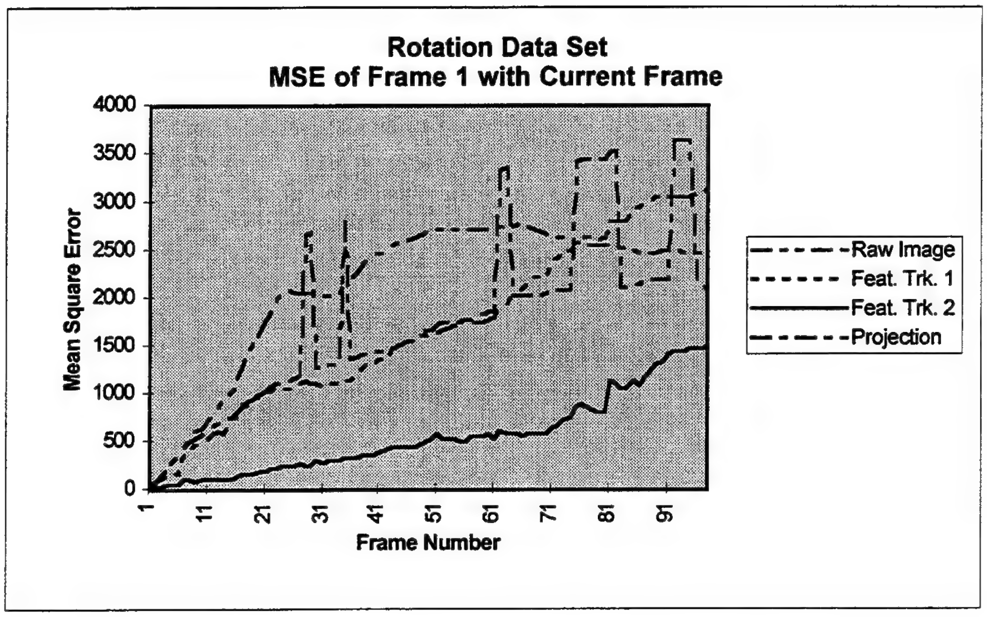

Figure 12: Real data set 2, MSE current frame to reference frame.

4. Conclusions

4.1 Projection Algorithm

The ARL projection algorithm represents the simplest of the stabilization algorithms evaluated. It can compensate for image translations to the nearest pixel. No subpixel matching or rotation correction is attempted. For both synthetic data set 1 and real data set 2, the algorithm performed well enough to allow the tracker to detect the majority of the targets with consistent tracks. The algorithm is unable to correct for rotations or scale changes, and therefore did not significantly stabilize real data set 2.

The largest drawback of this algorithm was the number of false alarms generated. These false alarms were present even with high tracker detection thresholds, showing that subpixel matching of the image sequence is necessary for proper tracker performance. Overall, it was found that this algorithm would be unacceptable as a front-end system for the tracker.

4.2 FTA 1

FTA 1 represents a significant increase in computational complexity over the projection algorithm. This algorithm was designed to operate on a moving platform and to eliminate high-frequency image movement due to rough terrain. As such, this algorithm stabilizes frame n to frame $n+1$. When this algorithm is used to evaluate the current frame against a slowly changing reference frame, extremely accurate subpixel matching must be present to ensure that errors do not propagate as the image sequence progresses. Unfortunately, real-world considerations, such as pixel blur due to camera motion, illumination changes, and perspective distortions, make such accurate matching nearly impossible. As shown in previous sections, this algorithm greatly reduces the mean square error of frame n relative to frame $n+1$ as compared to the raw sequence. In fact, the output image sequence looks very stable to the human eye. However, the drift that is present when one is viewing the reference frame compared to the current frame is significant enough that the tracker is unable to detect and track targets in the image sequences. The tracker performance shows that imagery that appears stable to a human observer may still have enough motion artifacts remaining that a computer does not accept the imagery as stable. Some of the error in stabilization may possibly be eliminated by a more sophisticated feature detection algorithm. The current feature detection scheme was found to pick features that lie on power lines (an object that makes feature rejection very difficult) and moving objects like tree limbs (causing the algorithm to compensate for tree motion instead of camera motion). Stable, unique features which exhibit sharp edges and high contrast would allow the feature tracking algorithm and stabilization to be more accurate. Overall, this algorithm would not be useful as a front end processor for the tracking system.

4.3 FTA 2

FTA 2 is based on FTA 1, with the addition of an extra step to realign each frame with the reference frame. As may be seen from Table 5, the performance difference between this algorithm and FTA 1 is very small. The average correction between the two algorithms is on the order of several hundredths of a pixel. However, this small difference is very important to the tracker. The false alarm rate drops to zero, and the percentage of targets detected increases to 100%. These figures clearly show the need for accurate subpixel matching in computer-viewed imagery.

Table 5: Average differences between FTA 2 and the other two algorithms.

	Row	Column	Rotation	Scale
FTA1–FTA2	0.38	0.23	0.10	0.00
Projection–FTA2	0.59	0.30	0.06	0.00

While operating on real data set 1, the tracker showed no significant performance loss over the stable data set. This performance shows that for unstable data rotations around the x and y axes, the extra computational cost of FTA 2 is necessary for proper operation. When working against data which involved rotations around all three axes (up to $\cong$ 45 degrees of total rotation around the z axis), the algorithm did not fare as well. Further investigation is necessary to determine the cause of the algorithm's breakdown, and exactly what range of rotation the algorithm is capable of compensating for. Overall, this algorithm would make a very good front end pre-processor for a modified ARL system which would allow for the scanning of a large tracking area.

4.4 Additional Research

This paper has used only stabilization algorithms which have public domain simulations available. To be truly complete, commercial algorithms, such as the one presented in Appendix C, which are proprietary in nature, must also be studied.

This paper has used only three FLIR data sets to evaluate the algorithms. Additional data sets that provide scale changes as well as more combinations of instabilities need to be examined. In addition, the performance of the algorithms against imagery from other sensors (video, laser radar, synthetic aperture radar) needs to be characterized.

Appendix A. Algorithm Errors on Synthetic Data Set 1

	Feature Tracking Algorithm 1				Feature Tracking Algorithm 2				Projection Algorithm		
Frame #	Row	Col	Angle	Scale	Row	Col	Angle	Scale	Row	Col	Angle
1	0.01	0.01	0.00	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.00
2	0.07	0.16	0.07	0.00	0.02	0.21	0.08	0.00	0.10	0.00	0.00
3	0.65	0.12	0.12	0.00	0.05	0.11	0.09	0.00	0.30	0.00	0.00
4	1.53	0.12	0.06	0.00	0.12	0.10	0.04	0.00	0.70	0.00	0.00
5	2.38	0.23	0.05	0.00	0.27	0.29	0.02	0.00	0.00	0.00	0.00
6	2.61	0.38	0.03	0.00	0.25	0.03	0.05	0.00	1.00	0.00	0.00
7	2.95	0.35	0.04	0.00	0.10	0.06	0.01	0.00	1.00	0.00	0.00
8	3.29	0.32	0.07	0.00	0.07	0.15	0.05	0.00	1.00	0.00	0.00
9	3.89	0.40	0.21	0.00	0.14	0.16	0.01	0.00	8.00	0.00	0.00
10	4.24	0.42	0.25	-0.01	0.18	0.18	0.02	0.00	8.00	0.00	0.00
11	5.12	0.41	0.27	-0.01	0.19	0.32	0.05	0.00	7.00	0.00	0.00
12	5.59	0.49	0.19	-0.01	0.10	0.20	0.01	0.00	6.00	0.00	0.00
13	5.93	0.50	0.12	-0.01	0.18	0.12	0.04	0.00	11.00	0.00	0.00
14	6.27	0.57	0.13	-0.01	0.19	0.12	0.04	0.00	1.00	1.00	0.00
15	6.59	0.74	0.16	-0.01	0.22	0.17	0.05	0.00	2.00	1.00	0.00
16	6.21	0.02	0.01	-0.01	0.07	0.17	0.00	0.00	2.00	1.00	0.00
17	7.13	0.00	0.04	-0.01	0.18	0.19	0.03	0.00	2.00	1.00	0.00
18	7.80	0.08	0.00	-0.01	0.25	0.19	0.06	0.00	1.00	1.00	0.00
19	8.06	0.04	0.04	-0.01	0.15	0.10	0.04	0.00	2.00	9.00	0.00
20	8.55	0.41	0.11	-0.01	0.06	0.08	0.03	0.00	9.00	8.00	0.00
21	9.15	0.63	0.17	0.00	0.24	0.19	0.00	0.00	8.00	7.00	0.00
22	9.16	1.04	0.73	0.00	0.11	0.05	1.04	0.00	2.00	2.00	1.00
23	8.62	0.48	1.86	0.00	1.14	0.71	2.11	0.00	6.00	3.00	3.00
24	9.09	0.04	2.39	0.00	1.87	2.19	3.01	0.00	11.00	4.00	6.00
25	7.22	1.79	3.65	0.00	3.54	4.48	3.94	0.00	18.00	8.00	10.00
26	5.02	4.33	5.33	0.00	6.14	7.80	4.95	0.00	9.00	2.00	5.00
27	8.34	1.54	9.27	0.00	2.82	3.66	9.07	0.00	9.00	4.00	-4.00
28	17.56	1.45	6.98	-0.01	4.23	2.24	7.26	0.00	6.00	2.00	3.00
29	14.44	1.69	8.19	-0.01	1.57	2.53	8.31	0.00	19.00	8.00	11.00
30	10.16	6.48	8.47	0.00	5.76	8.51	9.19	0.00	4.00	2.00	2.00
31	15.43	1.93	8.88	0.00	0.00	1.19	10.26	0.00	25.00	1.00	-8.00
32	26.48	5.58	9.50	0.01	11.77	0.85	7.69	0.01	2.00	2.00	1.00
33	23.14	8.72	2.90	0.01	7.97	6.47	0.48	0.01	5.00	3.00	3.00
34	23.54	8.53	3.53	0.01	9.94	6.98	0.30	0.01	11.00	3.00	6.00
35	22.39	8.84	4.58	0.02	9.22	8.07	1.02	0.02	22.00	3.00	10.00
36	19.97	9.39	4.29	0.02	6.22	9.81	8.11	0.03	12.00	2.00	5.00

Table units are pixels for row and column shifts, degrees for angle shifts, and percent for scale changes.

The row, column, and angle values in the table in Appendix A were computed by determining the absolute value of the difference between the actual image motion (as given in Table 1) and the motion reported by the various algorithms. The scale value is the actual scale value (100% for the entire sequence) minus the reported scale value.

Appendix B. Difference Between FTA 1 and FTA 2

Frame	Row	Column	Rotation	Scale	Frame	Row	Column	Rotation	Scale
1	0.00	0.00	0.00	100.00%	51	-0.49	-0.01	0.33	99.70%
2	-0.03	0.17	-0.03	99.93%	52	-0.67	0.19	-0.05	99.88%
3	-0.49	0.16	0.03	99.57%	53	-0.82	0.16	-0.03	100.64%
4	0.47	-0.31	-0.08	100.28%	54	0.60	-0.37	-0.06	99.50%
5	-0.23	-0.13	0.07	100.06%	55	-0.13	0.57	-0.07	100.05%
6	-0.30	0.17	0.02	99.73%	56	-0.21	0.38	-0.01	100.07%
7	0.38	-0.04	0.05	100.03%	57	-0.43	-0.09	0.03	100.22%
8	0.30	0.14	-0.16	100.33%	58	0.16	-0.43	0.14	100.25%
9	-0.25	-0.10	-0.01	99.80%	59	0.25	-0.41	0.42	99.63%
10	-0.31	0.16	0.12	100.08%	60	-0.16	0.32	-0.07	100.20%
11	0.57	-0.03	-0.14	100.01%	61	0.86	0.06	0.04	100.44%
12	0.05	0.13	0.06	99.92%	62	1.06	-0.10	0.09	99.89%
13	-0.44	-0.12	0.09	99.66%	63	0.81	-0.38	0.20	99.48%
14	-0.53	0.19	0.01	99.81%	64	0.77	-0.11	-0.08	100.07%
15	0.06	0.03	-0.22	99.99%	65	0.39	-0.40	0.09	99.60%
16	-0.39	-0.17	-0.14	99.83%	66	0.67	-0.08	0.21	100.39%
17	-0.13	0.17	-0.20	99.92%	67	1.05	-0.27	0.16	100.32%
18	-0.16	-0.53	0.10	99.44%	68	-0.14	0.27	-0.16	99.93%
19	-0.67	0.37	-0.08	100.23%	69	0.40	0.96	-0.12	99.81%
20	-1.90	1.54	-0.06	100.73%	70	0.09	0.26	0.10	100.09%
21	-1.37	1.77	0.05	101.03%	71	-0.81	-0.18	0.10	99.60%
22	0.28	-0.02	0.01	100.06%	72	-0.33	0.12	-0.09	100.10%
23	0.34	0.16	0.37	99.75%	73	0.91	-0.03	0.32	100.38%
24	0.13	-0.21	-0.10	99.99%	74	1.07	0.06	0.17	100.64%
25	0.11	0.23	0.05	100.47%	75	0.36	0.11	0.17	100.20%
26	0.12	0.14	0.12	100.11%	76	0.46	0.30	-0.09	100.00%
27	0.52	-0.23	0.11	100.42%	77	0.05	-0.16	-0.08	100.10%
28	-0.12	0.17	-0.03	100.03%	78	-0.32	0.02	0.22	99.75%
29	-0.05	-0.22	-0.09	100.10%	79	0.57	-0.35	0.11	99.81%
30	-0.30	0.17	0.15	100.11%	80	-0.20	0.02	-0.18	99.94%
31	0.16	0.04	-0.06	99.64%	81	0.07	0.12	-0.04	100.20%
32	0.18	0.35	-0.16	99.91%	82	0.00	-0.06	0.14	99.87%
33	0.07	0.23	-0.15	100.05%	83	0.21	-0.13	0.02	100.01%
34	0.49	-0.12	0.10	99.95%	84	0.27	-0.01	-0.16	100.27%
35	-0.61	0.41	-0.05	99.92%	85	0.62	-0.04	-0.01	100.05%
36	0.25	0.06	-0.04	100.37%	86	0.26	-0.45	0.22	99.87%
37	0.03	-0.36	0.08	100.01%	87	0.06	0.32	0.00	100.05%
38	-0.18	0.35	0.04	99.90%	88	0.58	0.09	-0.18	100.12%
39	0.55	-0.19	0.00	99.91%	89	-0.09	-0.11	-0.10	99.88%
40	-0.68	0.28	-0.06	99.97%	90	0.21	0.16	-0.01	100.10%
41	0.07	0.12	0.13	99.76%	91	0.44	-0.74	0.14	99.88%
42	-0.06	-0.03	0.08	100.06%	92	0.33	0.08	-0.05	100.08%
43	0.14	-0.35	-0.05	100.34%	93	0.65	-0.06	0.09	100.09%
44	-0.33	0.22	-0.13	99.95%	94	0.13	0.36	-0.03	100.15%
45	-0.37	0.47	0.01	100.06%	95	0.26	-0.22	-0.12	99.86%
46	-0.27	0.05	0.08	99.68%	96	-0.13	0.02	-0.01	100.20%
47	0.35	-0.16	0.08	99.75%	97	-0.28	0.17	0.01	99.78%
48	0.30	-0.22	-0.01	100.28%	98	0.30	0.14	0.00	100.08%
49	0.45	0.59	-0.12	99.92%	99	-0.04	-0.24	-0.01	99.66%
50	0.74	-0.19	0.02	100.30%	100	0.09	0.06	0.05	100.08%

Rows and columns are in pixels, angle is in degrees, and scale is percent of FTA 1 scale.

The values in the table in Appendix B represent the additional changes made to each image by the FTA 2 algorithm over the FTA 1 algorithm. For example, frame 2 was shifted an additional −0.03 pixels in row and an additional 0.17 pixels in column, and was rotated an additional −0.03 degrees over the FTA 1 output. In addition to these corrections for frame 2, the FTA 2 algorithm reduced its output to be 99.93% of the scale of the FTA 1 output.

Appendix C. Additional Stabilization Algorithm

An additional commercial stabilization algorithm was developed by the David Sarnoff Research Center. This algorithm is an innovative extension to an optical flow technique which uses image pyramids to estimate motion regions in images. An assumption fundamental to most optical flow algorithms is that motion at any one point may be represented by a simple translation [17, 20]. It is assumed that even a complex motion will appear as a uniform translation when viewed through a sufficiently small window. Bergen et al. [6] found that this assumption was not valid along the boundary between two differently moving image regions.

Bergen proposed an alternative formulation to the traditional single motion component optical flow algorithm [10, 21, 22, 23]. This algorithm allows for two distinct patterns to be undergoing affine motion within a given local analysis region. The algorithm is iterative, estimating a region's motion, then removing that region through a nulling procedure. This allows for a more precise estimation of the remaining motion components. This algorithm runs in simulation on Sun computers, and there are currently plans to implement this set of algorithms on Sarnoff's custom VFE hardware or on a new custom hardware set.

Multi-motion Algorithm

The Sarnoff algorithm assumes that the image frame contains two independent patterns ($P(x,y)$ and $Q(x,y)$) which have independent motions of $\boldsymbol{p}(x,y)$ and $\boldsymbol{q}(x,y)$. If a direct extension of single motion estimation is attempted, it becomes necessary to first estimate the derivatives of both patterns. The problem arises in that in order to estimate these derivatives, the patterns must first be separated or segmented. The Sarnoff approach eliminates the need to segment the image prior to simultaneously estimating the two motion components.

Two-Component Motion Model

The Sarnoff two-component motion model assumes that within a region R, the image I may be represented by a combined function of $P(x,y)$ and $Q(x,y)$:

$$I(x,y,0) = P(x,y) \oplus Q(x,y)$$

and

$$I(x,y,t) = P^{t\mathbf{p}} \oplus Q^{t\mathbf{q}} \tag{C.1}$$

where $p^{t\mathbf{p}}$ denotes the pattern P transformed by the motion $t\mathbf{p}$.

The symbol $\oplus$ represents an operator such as addition or multiplication that combines the two patterns. For example, in the case of the boundary between two motion patterns, the region may be represented by the sum of two patterns that are defined over the entire analysis region, but which have zero amplitude over complementary portions of the region.

Estimating Two Motions

The key observation in the Sarnoff approach is that if one of the motion components ($\mathbf{p}$ or $\mathbf{q}$) and the combination rule ($\oplus$) are known, then that motion component may be removed. The image motion may then be solved by the traditional one-motion algorithm without determining the actual patterns (P and Q). We will assume that the combination rule is addition.

Let us assume that the motion $\mathbf{p}$ is known, and that we must determine the motion $\mathbf{q}$. The pattern P, moving at velocity $\mathbf{p}$, may be removed by shifting each frame by $\mathbf{p}\Delta t$ and subtracting it from the following frame. The resulting sequence will be void of any contribution from pattern P and may be assumed to contain only patterns moving with velocity $\mathbf{q}$. This difference image sequence may be defined by

$$\begin{aligned} D_t &\equiv I(x, y, t+1) - I^{\mathbf{p}}(x, y, t) \\ &= \left(P^{(t+1)\mathbf{p}} + Q^{(t+1)\mathbf{q}}\right) - \left(P^{(t+1)\mathbf{p}} + Q^{t\mathbf{q}+\mathbf{p}}\right) \\ &= \left(Q^{\mathbf{q}} - Q^{\mathbf{p}}\right)^{t\mathbf{q}}. \end{aligned} \tag{C.2}$$

As is shown in equation (C.2), the sequence $\{D_t\}$ may now be represented by the new pattern $(Q^{\mathbf{q}} - Q^{\mathbf{p}})$ moving with a single motion velocity $\mathbf{q}$. We may therefore solve for the image motion using the traditional single-motion model. Note that this procedure removes one of the patterns from the image sequence without explicitly determining what that pattern is.

In the real world, the motions $\mathbf{p}$ and $\mathbf{q}$ are not normally known. In fact, estimates of these motions may not even exist. However, by using an alternating iterative refinement procedure, both motions may be recovered. This is true even if initial estimates of $\mathbf{p}=0$ and $\mathbf{q}=0$ are used. In this procedure, estimates alternate between $\mathbf{p}$ and $\mathbf{q}$. Therefore, $\mathbf{p}$ is obtained on even-numbered cycles while $\mathbf{q}$ is obtained on odd cycles. These iterative cycles are repeated until $\Delta\mathbf{p} = \mathbf{p}_n - \mathbf{p}_{n+1}$ and $\Delta\mathbf{q} = \mathbf{q}_n - \mathbf{q}_{n+1}$ are sufficiently small.

Although this algorithm runs in simulation on a Sun computer platform, it was not available for characterization.

References

1 Webster's New World Dictionary, Third College Edition, Copyright 1988 by Simon & Schuster, Inc.

2 S. Balakirsky, P. David, et al., "Semi-autonomous mobile target engagement system," *in Proc. Symposium of the Association for Unmanned Vehicle Systems*, 1993.

3 T. Huang and R. Tsai, *Image Sequence Analysis*, Springer-Verlag, Berlin, 1981.

4 Q. Zheng and R. Chellappa, "A novel image registration algorithm," in *Proc. DARPA Image Understanding Workshop* (San Diego, CA), pp. 899–909, Jan. 1992.

5 C. Morimoto et al., "Detection of independently moving objects in passive video," in *Proc. Intelligent Vehicles Symposium* (Detroit, MI), pp. 270–275, Sept. 1995.

6 J. Bergen, P. Burt, R. Hingorani, and S. Peleg, "A three-frame algorithm for estimating two-component image motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 886–896, 1992.

7 P. J. Burt, "Fast filter transforms for image processing," *Computer Graphics Image Processing*, vol. 16, pp. 20–51, 1981.

8 P. Anandan, "A unified perspective on computational techniques for the measurement of visual motion," in *Proc. International Conference on Computer Vision* (London, England), pp. 219–230, May 1987.

9 J. R. Bergen and E. H. Adelson, "Hierarchical, computationally efficient motion estimation algorithm," *J. Opt. Soc. Amer. A.*, vol. 4, p. 35, 1987.

10 D. J. Heeger, "Optical flow using spatiotemporal filters," *Int. J. Computer Vision*, vol. 1, pp. 279–302, 1988.

11 A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, Englewood Cliffs, NJ, 1989, pp. 347–357.

12 Q. Zheng and R. Chellappa, "Automatic feature point extraction and tracking in image sequences for arbitrary camera motion," University of Maryland Technical Report CAR-TR-628, June 1992.

13 Q. Tian and M. N. Huhns, "Algorithms for subpixel registration," *Computer Vision, Graphics, Image Processing*, vol. 35, pp. 220–233, 1986.

14 R. Deriche and O. Faugeras, "Tracking line segments," in *Proc. European Conference on Computer Vision* (Nice, France), pp. 259–268, 1990.

15 B. Bhanu, "Evaluation of automatic target recognition algorithms," *Proc. SPIE*, vol. 435, pp. 18–27, 1983.

16 D. Ballard and C. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

17 B. K. P. Horn, *Robot Vision*, MIT Press, Cambridge, MA, 1986.

18 Y. Bar-Shalom, "Tracking methods in a multitarget environment," *IEEE Transactions on Automatic Control*, vol. 23, pp. 618–626, 1978.

19 N. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.

20 C. L. Fennema and W. B. Thompson, "Velocity determination in scenes containing several moving objects," *Computer Graphics Image Processing*, vol. 9, pp. 301–315, 1979.

21 B. K. P. Horn and E. J. Weldon, "Direct methods for recovering motion," *Int. J. Computer Vision*, vol. 2, pp. 51–76, 1988.

22 J. O. Limb and J. A. Murphy, "Estimating the velocity of moving images in television signals," *Computer Graphics Image Processing*, vol. 4, pp. 311—327, 1975.

23 B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. DARPA Image Understanding Workshop*, 1981, pp. 121–130.